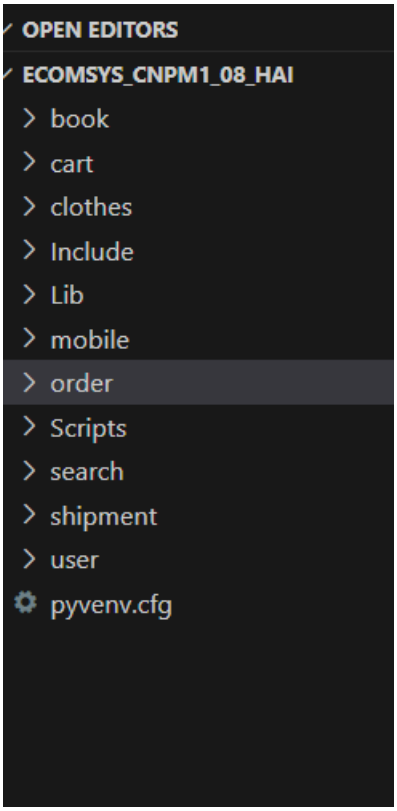
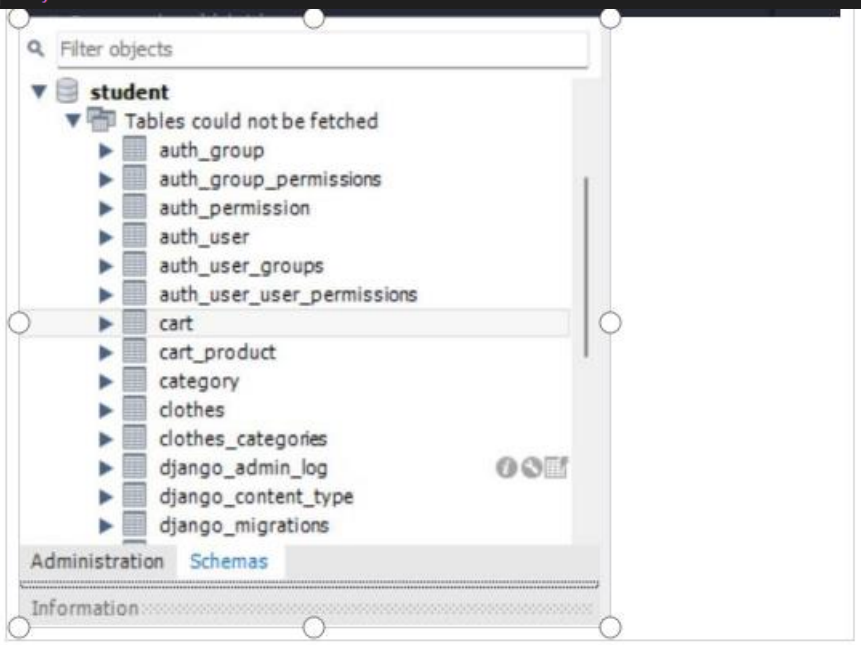
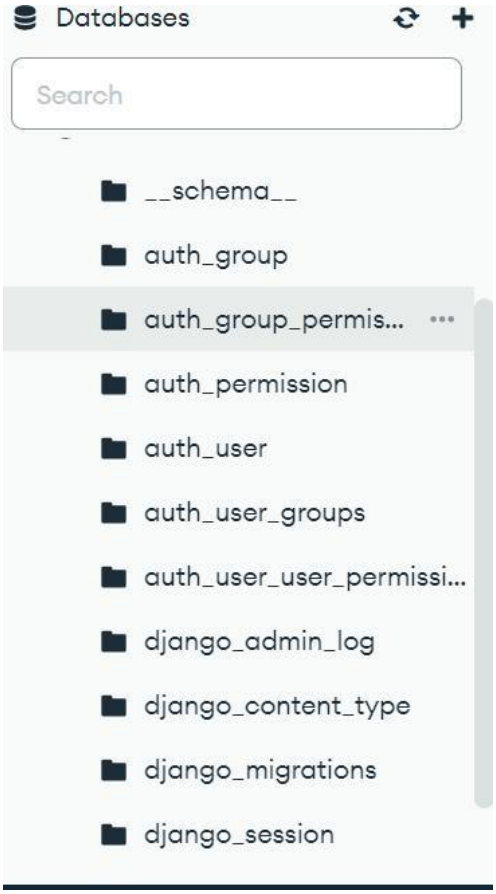
1.Create app : book, mobile, clothes, payment, shipment, order, search, category



2. Cơ sở dữ liệu

```python
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'shop',
        'USER': 'root',
        'PASSWORD': 'duchai258',
        'HOST': 'localhost',
        'PORT': '3306',
    },
    'mongodb': {
        'ENGINE': 'djongo',
        'CLIENT': {
            'host': 'mongodb+srv://bduchai:duchai258@cluster0.zyl6hxk.mongodb.net/?retryWrites=true&w=majority', # Repla
            'port': 27017,
        },
        'NAME': 'shop',
    }
```





2.Khai báo app trong setings.py

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'product',
    'cart',
    'category',
    'search',
    'shipment',
    'payment',
    'order',
    'mobile',
    'clothes',
    'book'
]
```

3.Config Database cho cả mongodb và mysqldb

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'shop',
        'USER': 'root',
        'PASSWORD': 'duchai258',
        'HOST': 'localhost',
        'PORT': '3306',
    },
    'mongodb': {
        'ENGINE': 'djongo',
        'CLIENT': {
            'host': 'mongodb+srv://bduchai:duchai258@cluster0.zyl6hxk.mongodb.net/?retryWrites=true&w=majority', # Repla
            'port': 27017,
        },
        'NAME': 'shop',
    }
```

4.Tạo Router để điều hướng dữ liệu tới database

```
class MySqlRouter:
    def db_for_read(self, model, **hints):
        if model._meta.app_label == 'cart'or model._meta.app_label == 'shipment' or model._meta.app_label == 'order':
            return 'mysql_db'
        elif model._meta.app_label == 'product' or model._meta.app_label == 'category' or model._meta.app_label == 'mobile':
            return 'default'
        # return 'default'

    def db_for_write(self, model, **hints):
        if model._meta.app_label == 'cart':
            return 'mysql_db'
        elif model._meta.app_label == 'product' or model._meta.app_label == 'category' or model._meta.app_label == 'mobile' :
            return 'default'
        # return 'default'

    def allow_relation(self, obj1, obj2, **hints):
        if obj1._meta.app_label == 'cart' or obj2._meta.app_label == 'cart':
            return True
        elif 'cart' not in [obj1._meta.app_label, obj2._meta.app_label]:
            return True
        elif obj1._meta.app_label == 'product' or obj2._meta.app_label == 'product':
            return True
        elif 'userchecking' not in [obj1._meta.app_label, obj2._meta.app_label]:
            return True
        elif obj1._meta.app_label == 'category' or obj2._meta.app_label == 'category':
            return True
        elif obj1._meta.app_label == 'mobile' or obj2._meta.app_label == 'mobile':
            return True
        elif 'category' not in [obj1._meta.app_label, obj2._meta.app_label]:
            return True
        return False

    def allow_migrate(self, db, app_label, model_name=None, **hints):
        if app_label == 'cart' or app_label == 'shipment' or app_label == 'payment' or app_label == 'order':
            return db == 'mysql_db'
```

5Code model.py , views.py , urls.py cho app Book

5.1 Model.py

```python
class Book(models.Model):
    category = models.ForeignKey(Category_Book, on_delete=models.CASCADE,null=True)
    title = models.CharField(max_length=255)
    author = models.CharField(max_length=255)
    description = models.TextField()
    price = models.DecimalField(default=0.0,max_digits=10, decimal_places=2)
    created_at = models.DateTimeField(default=timezone.now)
    updated_at = models.DateTimeField(auto_now=True)
```

5.2 views.py

```python
# Create your views here.
@api_view(['POST']) # neu khong co se bi error : 403 Forbidden
def create_book(request):
    serializer = BookSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

@api_view(['GET'])
def get_all(request):
        books = Book.objects.all()
        serializer = BookSerializer(books, many=True)
        return Response(serializer.data)
```

```python
@api_view(['GET'])
def get_byId(request,book_id):
        try:
            category = BookSerializer(Book.objects.get(pk=book_id)) # this is used when you want to convert a r
            # category = CategorySerializer(data=Category.objects.get(pk=category_id)) #his usage suggests that
            return Response(category.data)
        except Book.DoesNotExist:
            return Response('book does not exist', status=status.HTTP_400_BAD_REQUEST)

@api_view(['DELETE'])
def delete_book(request,book_id):
        book = Book.objects.get(pk=book_id)
        if not book:
            return Response(f'book with id={book_id} not exist',status=status.HTTP_404_NOT_FOUND)
        book.delete()
        return Response(f"Successfully deleted book with id={book_id}",status=status.HTTP_204_NO_CONTENT)
```

```python
@api_view(['PUT'])
def update_book(request, book_id):
    book = Book.objects.get(pk=book_id)
    if not book:
        print('not founddd')
        return Response(status=status.HTTP_404_NOT_FOUND)
    serializer = BookSerializer(book, data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

5.3 urls.py

```
urlpatterns = [
    path('book/get_all', get_all, name='book_list'),
    path('book/detail/<str:book_id>', get_byId, name='book_detail'),
    path('book/create',create_book, name='create_book'),
    path('book/update/<str:book_id>', update_book, name='update_book'),
    path('book/delete/<str:book_id>', delete_book, name='delete_book'),
```

6. Code model.py , views.py , urls.py cho

a

```
class Clothes(models.Model):
    category = models.ForeignKey(Category_Clothes, on_delete=models.CASCADE, null=True)
    title = models.CharField(max_length=255)
    brand = models.CharField(max_length=255)
    description = models.TextField()
    price = models.DecimalField(default=0.0, max_digits=10, decimal_places=2)
    created_at = models.DateTimeField(default=timezone.now)
    updated_at = models.DateTimeField(auto_now=True)
```

6.2 views.py

```python
@api_view(['POST']) # neu khong co se bi error : 403 Forbidden
def create_clothes(request):
    serializer = ClothesSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

@api_view(['GET'])
def get_all_clothes(request):
        clothes = Clothes.objects.all()
        serializer = ClothesSerializer(clothes, many=True)
        return Response(serializer.data)


@api_view(['GET'])
def get_byId_clothes(request,clothes_id):
        try:
            clothes = ClothesSerializer(Clothes.objects.get(pk=clothes_id)) # this is used wher
            # category = CategorySerializer(data=Category.objects.get(pk=category_id)) #his usa
            return Response(clothes.data)
        except Clothes.DoesNotExist:
            return Response('book does not exist', status=status.HTTP_400_BAD_REQUEST)
```

```python
@api_view(['DELETE'])
def delete_clothes(request,clothes_id):
        clothes = Clothes.objects.get(pk=clothes_id)
        if not clothes:
            return Response(f'book with id={clothes_id} not exist',status=status.HTTP_404_NOT_FOUND)
        clothes.delete()
        return Response(f"Successfully deleted book with id={clothes_id}",status=status.HTTP_204_NO_CONTENT)

@api_view(['PUT'])
def update_clothes(request, clothes_id):
    clothes = Clothes.objects.get(pk=clothes_id)
    if not clothes:
        print('not founddd')
        return Response(status=status.HTTP_404_NOT_FOUND)
    serializer = ClothesSerializer(clothes, data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

6.3 urls.py

```python
  path('clothes/get_all', get_all_clothes, name='clothes_list'),
  path('clothes/detail/<str:clothes_id>', get_byId_clothes, name='clothes_detail'),
  path('clothes/create',create_clothes, name='create_clothes'),
  path('clothes/update/<str:clothes_id>', update_clothes, name='update_clothes'),
  path('clothes/delete/<str:clothes_id>', delete_clothes, name='delete_clothes'),
```

7.Code model.py , views.py , urls.py cho app Clothes

7.1 model.py

```python
class Mobile(models.Model):
    brand = models.CharField(max_length=255)
    price = models.DecimalField(default=0.0, max_digits=10, decimal_places=2)
    description = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```

7.2 views.py

```python
@api_view(['POST']) # neu khong co se bi error : 403 Forbidden
def create_clothes(request):
    serializer = ClothesSerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

@api_view(['GET'])
def get_all_clothes(request):
        clothes = Clothes.objects.all()
        serializer = ClothesSerializer(clothes, many=True)
        return Response(serializer.data)


@api_view(['GET'])
def get_byId_clothes(request,clothes_id):
        try:
            clothes = ClothesSerializer(Clothes.objects.get(pk=clothes_id)) # this is used wher
            # category = CategorySerializer(data=Category.objects.get(pk=category_id)) #his usa
            return Response(clothes.data)
        except Clothes.DoesNotExist:
            return Response('book does not exist', status=status.HTTP_400_BAD_REQUEST)
```

```python
@api_view(['DELETE'])
def delete_clothes(request,clothes_id):
        clothes = Clothes.objects.get(pk=clothes_id)
        if not clothes:
            return Response(f'book with id={clothes_id} not exist',status=status.HTTP_404_NOT_FOUND)
        clothes.delete()
        return Response(f"Successfully deleted book with id={clothes_id}",status=status.HTTP_204_NO_CONTENT)

@api_view(['PUT'])
def update_clothes(request, clothes_id):
    clothes = Clothes.objects.get(pk=clothes_id)
    if not clothes:
        print('not founddd')
        return Response(status=status.HTTP_404_NOT_FOUND)
    serializer = ClothesSerializer(clothes, data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

7.3 urls.py

```
urlpatterns = [
    path('Mobile/get_all', get_all, name='Mobile_list'),
    path('Mobile/detail/<str:Mobile_id>', get_byId, name='Mobile_detail'),
    path('Mobile/create',create_Mobile, name='create_Mobile'),
    path('Mobile/update/<str:Mobile_id>', update_Mobile, name='update_Mobile'),
    path('Mobile/delete/<str:Mobile_id>', delete_Mobile, name='delete_Mobile'),
```

8. Code model.py , views.py , urls.py cho app Category

8.1 model.py

```
# Create your models here.
class Category_Book(models.Model):
    name = models.CharField(max_length=255)
    description = models.CharField(max_length=255)

    def __str__(self):
            return self.name

class Category_Clothes(models.Model):
    name = models.CharField(max_length=255)
    description = models.CharField(max_length=255)

    def __str__(self):
            return self.name

class Category_Mobile(models.Model):
    name = models.CharField(max_length=255)
    description = models.CharField(max_length=255)

    def __str__(self):
            return self.name
```

8.2 views.py

```
@api_view(['POST']) # neu khong co se bi error : 403 Forbidden
def create_category_book(request):
    serializer = CategorySerializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

@api_view(['GET'])
def get_all_book(request):
        books = Category_Book.objects.all()
        serializer = CategorySerializer(books, many=True)
        return Response(serializer.data)
```

```python
@api_view(['GET'])
def get_byId_book(request,category_id):
        try:
            category = CategorySerializer(Category_Book.objects.get(pk=category_id)) # this is used when you want to
            # category = CategorySerializer(data=Category.objects.get(pk=category_id)) #his usage suggests that the
            return Response(category.data)
        except Category_Book.DoesNotExist:
            return Response('Category does not exist', status=status.HTTP_400_BAD_REQUEST)

@api_view(['DELETE'])
def delete_category_book(request,category_id):
        category = Category_Book.objects.get(pk=category_id)
        if not category:
            return Response(f'category with id={category_id} not exist',status=status.HTTP_404_NOT_FOUND)
        category.delete()
        return Response(f"Successfully deleted category with id={category_id}",status=status.HTTP_204_NO_CONTENT)
```

```python
@api_view(['PUT'])
def update_category_book(request, category_id):
    category = Category_Book.objects.get(pk=category_id)
    if not category:
        print('not founddd')
        return Response(status=status.HTTP_404_NOT_FOUND)
    serializer = CategorySerializer(category, data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

```python
# Create your views here.
@api_view(['POST']) # neu khong co se bi error : 403 Forbidden
def create_category_clothes(request):
    serializer = Category_Clothes_Serializer(data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data, status=status.HTTP_201_CREATED)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

@api_view(['GET'])
def get_all_category_clothes(request):
        books = Category_Clothes.objects.all()
        serializer = Category_Clothes_Serializer(books, many=True)
        return Response(serializer.data)
```

```python
@api_view(['GET'])
def get_byId_category_clothes(request,category_id):
        try:
            category = Category_Clothes(Category_Clothes.objects.get(pk=category_id)) # this is used when you want
            # category = CategorySerializer(data=Category.objects.get(pk=category_id)) #his usage suggests that the
            return Response(category.data)
        except Category_Clothes.DoesNotExist:
            return Response('Category does not exist', status=status.HTTP_400_BAD_REQUEST)

@api_view(['DELETE'])
def delete_category_clothes(request,category_id):
        category = Category_Clothes.objects.get(pk=category_id)
        if not category:
            return Response(f'category with id={category_id} not exist',status=status.HTTP_404_NOT_FOUND)
        category.delete()
        return Response(f"Successfully deleted category with id={category_id}",status=status.HTTP_204_NO_CONTENT)
```

```python
@api_view(['PUT'])
def update_category_clothes(request, category_id):
    category = Category_Clothes.objects.get(pk=category_id)
    if not category:
        print('not founddd')
        return Response(status=status.HTTP_404_NOT_FOUND)
    serializer = Category_Clothes_Serializer(category, data=request.data)
    if serializer.is_valid():
        serializer.save()
        return Response(serializer.data)
    return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

8.3 urls.py

```python
urlpatterns = [
    path('category_book/get_all', get_all_book, name='category_list_book'),
    path('category_book/detail/<str:category_id>', get_byId_book, name='category_detail_book'),
    path('category_book/create',create_category_book, name='create_category_book'),
    path('category_book/update/<str:category_id>', update_category_book, name='update_category_book'),
    path('category_book/delete/<str:category_id>', delete_category_book, name='delete_category_book'),


    path('category_clothes/get_all', get_all_category_clothes, name='category_list_clothes'),
    path('category_clothes/detail/<str:category_id>', get_byId_category_clothes, name='category_detail_clothes'),
    path('category_clothes/create',create_category_clothes, name='create_category'),
    path('category_clothes/update/<str:category_id>', update_category_clothes, name='update_category_clothes'),
    path('category_clothes/delete/<str:category_id>', delete_category_clothes, name='delete_category_clothes'),

]
```

9. iews.py trong app search

```python
@api_view(['GET'])
def search_book(request):
        # query = request.GET.get('title') #lấy dữ liệu từ string query
        query = request.data.get('title') # lấy dữ liệu từ payload
        print(query)
        if query:
            books = Book.objects.filter(title__icontains=query)
            serializer = BookSerializer(books, many=True)
            return Response(serializer.data)
        else:
            return Response('Book does not exist', status=status.HTTP_400_BAD_REQUEST)
```

```python
urlpatterns = [
    path('book/search', search_book, name='search_book'),
]
```

10. Code model.py , views.py , urls.py cho app Cart

10.1 model.py

```python
# Create your models here.
class Cart(models.Model):
    user_id = models.CharField(max_length=24)
    quantity = models.PositiveIntegerField(default=1)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"{self.user.username}'s Cart"

class CartItem(models.Model):
    product_id = models.CharField(max_length=24)
    cart = models.ForeignKey(Cart, on_delete=models.CASCADE)
    #Cart: This is the model class that the foreign key is referring to. In this case, Cart is the target model, and
    #on_delete=models.CASCADE: This parameter specifies the behavior to follow when the referenced Cart instance is
```

10.2 views.py

```python
# Create your views here.
@api_view(['POST'])
def add_book_to_cart(request):
    try:
        user_id = request.data.get('user_id')
        book_id = request.data.get('book_id')

        # Kiểm tra xem trong bảng Cart đã tồn tại cart của người dùng hay chưa
        try:
            cart = Cart.objects.get(user_id=user_id)
            cart.quantity += 1
            cart.save()
            cart_item = CartItem.objects.create(product_id=book_id, cart=cart)
            created = False
        except Cart.DoesNotExist:
            # Tạo mới cart nếu chưa tồn tại
            print(222222222)
            cart = Cart.objects.create(user_id=user_id, quantity=1)
            cart_item = CartItem.objects.create(product_id=book_id, cart=cart)
            created = True

        print(user_id+"  "+book_id)
        if created:
            message = 'Cart created and book added successfully'
        else:
            message = 'Book added to the cart successfully'

        return Response({'message': message})
    except Exception as e:
        return Response({'error': str(e)}, status=400)
```

10.3 urls.py

```python
urlpatterns = [
    path('cart/add_book_to_cart', add_book_to_cart, name='add_book_to_cart'),
]
```

11. Code model.py , views.py , urls.py cho app shipment

11.1 model.py

```
# Create your models here.
class shipment(models.Model):
    ### The following are the fields of our table.
    fname = models.CharField(max_length=50)
    lname = models.CharField(max_length=50)
    email = models.CharField(max_length=50)
    mobile = models.CharField(max_length=12)
    address = models.CharField(max_length=200)
    product_id = models.CharField(max_length=10)
    quantity = models.CharField(max_length=5)
    payment_status = models.CharField(max_length=15)
    transaction_id = models.CharField(max_length=5)
    shipment_status = models.CharField(max_length=20)
    ### It will help to print the values.
    def __str__(self):
        return '%s %s %s %s %s %s %s %s %s %s' % (self.fname, self.
        lname, self.email, self.mobile, self.product_id, self.address, self.
        quantity , self.payment_status, self.transaction_id, self.shipment_status)
```

11.2 views.py

```
### This function is inserting the data into our table.
def ship_data_insert(fname, lname, email, mobile, address, product_id,
quantity, payment_status, transaction_id, shipment_status):
    shipment_data = ship_obj(fname = fname,lname = lname, email = email,
    mobile = mobile,
    address = address, product_id = product_id, quantity = quantity,
    payment_status = payment_status, transaction_id = transaction_id,
    shipment_status = shipment_status)
    shipment_data.save()
    return 1
```

```python
### This function will get the data from the front end.
@csrf_exempt
def shipment_reg_update(request):
    if request.method == 'POST':
        if 'application/json' in request.META['CONTENT_TYPE']:
            val1 = json.loads(request.body)
            ### This is for reading the inputs from JSON.
            fname = val1.get("First Name")
            lname = val1.get("Last Name")
            email = val1.get("Email Id")
            mobile = val1.get("Mobile Number")
            address = val1.get("Address")
            product_id = val1.get("Product Id")
            quantity = val1.get("Quantity")
            payment_status = val1.get("Payment Status")
            transaction_id = val1.get("Transaction Id")
            shipment_status = "ready to dispatch"

            resp = {}
            ### After all validation, it will call the data_insert function.
            respdata = ship_data_insert(fname, lname, email, mobile,
            address, product_id, quantity, payment_status, transaction_id, shipment_status)
        ### If it returns value then will show success.
            if respdata:
                resp['status'] = 'Success'
                resp['status_code'] = '200'
                resp['message'] = 'Product is ready to dispatch.'
            ### If value is not found then it will give failed in response.
            else:
                resp['status'] = 'Failed'
                resp['status_code'] = '400'
                resp['message'] = 'Failed to update shipment details.'
    return HttpResponse(json.dumps(resp), content_type = 'application/json')
```

```python
### This function is used for finding the transaction.
def shipment_data(uname):
    data = ship_obj.objects.filter(email = uname)
    for val in data.values():
        return val


### This function is used for getting the shipment status
@csrf_exempt
def shipment_status(request):
    if request.method == 'POST':
        if 'application/json' in request.META['CONTENT_TYPE']:
            variable1 = json.loads(request.body)
            ### This is for reading the inputs from JSON.
            uname = variable1.get("User Name")
            resp = {}
            ### It will call the shipment_data function.
            respdata = shipment_data(uname)
            ### If it returns value then will show success.
            if respdata:
                resp['status'] = 'Success'
                resp['status_code'] = '200'
                resp['message'] = respdata
                ### If it is not returning any value then it will show failed response.
            else:
                resp['status'] = 'Failed'
                resp['status_code'] = '400'
                resp['message'] = 'User data is not available.'
    return HttpResponse(json.dumps(resp), content_type = 'application/json')
```

11.3 urls.py

```python
urlpatterns = [
    path('api/shipment/shipment_status', shipment_status, name='shipment_status'),
    path('api/shipment/shipment_updates', shipment_reg_update, name='shipment_reg_update'),

]
```

12. Code model.py , views.py , urls.py cho app Payment

12.1 Model

```
# This is our model for user registration.
class payment_status(models.Model):
    ### The following are the fields of our table.
    username = models.CharField(max_length=10)
    product_id = models.CharField(max_length=10)
    price = models.CharField(max_length=10)
    quantity = models.CharField(max_length=5)
    mode_of_payment = models.CharField(max_length=20)
    mobile = models.CharField(max_length=12)
    status = models.CharField(max_length=15)
    ### It will help to print the values.
    def __str__(self):
        return '%s %s %s %s %s %s %s ' % (self.username, self.product_id, self.price, self.quantity,
```

12.2 views.py

```
### This function is for fetching the user data.
def get_transaction_details(uname):
    user = paystat.objects.filter(username = uname)
    for data in user.values():
        return data

### This function is used for storing the data.
def store_data(uname, prodid, price, quantity, mode_of_payment, mobile):
    user_data = paystat(username = uname, product_id = prodid, price =
    price, quantity = quantity, mode_of_payment = mode_of_payment, mobile =
    mobile, status = "Success")

    user_data.save()
    return 1
```

```python
### This function is created for getting the payment.
@csrf_exempt
def get_payment(request):
    uname = request.POST.get("User name")
    prodid = request.POST.get("Product id")
    price = request.POST.get("Product price")
    quantity = request.POST.get("Product quantity")
    mode_of_payment = request.POST.get("Payment mode")
    mobile = request.POST.get("Mobile Number")
    print(uname)
    print(prodid)
    print(price)
    print(quantity)
    print(mode_of_payment)
    print(mobile)
    resp = {}
    if uname and prodid and price and quantity and mode_of_payment and  mobile:
        ### It will call the store data function.
        respdata = store_data(uname, prodid, price, quantity, mode_of_payment, mobile)
        respdata2 = ship_update(uname)
        ### If it returns value then will show success.
        if respdata:
            resp['status'] = 'Success'
            resp['status_code'] = '200'
            resp['message'] = 'Transaction is completed.'
        ### If it is returning null value then it will show failed.
        else:
            resp['status'] = 'Failed'
            resp['status_code'] = '400'
            resp['message'] = 'Transaction is failed, Please try again.'
        ### If any mandatory field is missing then it will be through a failed message.
    else:
        resp['status'] = 'Failed'
        resp['status_code'] = '400'
        resp['message'] = 'All fields are mandatory.'
    return HttpResponse(json.dumps(resp), content_type = 'application/json')
```

```python
### This function is created for getting the username and password.
@csrf_exempt
def user_transaction_info(request):
    # uname = request.POST.get("User Name")
    if request.method == 'POST':
        if 'application/json' in request.META['CONTENT_TYPE']:
            val1 = json.loads(request.body)
            uname = val1.get('User Name')
            resp = {}
            if uname:
                ## Calling the getting the user info.
                respdata = get_transaction_details(uname)
                if respdata:
                    resp['status'] = 'Success'
                    resp['status_code'] = '200'
                    resp['data'] = respdata
                 ### If a user is not found then it give failed as response.
                else:
                    resp['status'] = 'Failed'
                    resp['status_code'] = '400'
                    resp['message'] = 'User Not Found.'
            ### The field value is missing.
            else:
                resp['status'] = 'Failed'
                resp['status_code'] = '400'
                resp['message'] = 'Fields is mandatory.'
        else:
            resp['status'] = 'Failed'
            resp['status_code'] = '400'
            resp['message'] = 'Request type is not matched.'
    else:
        resp['status'] = 'Failed'
        resp['status_code'] = '400'
        resp['message'] = 'Request type is not matched.'
    return HttpResponse(json.dumps(resp), content_type = 'application/son')
```

12.3 urls.py

```
urlpatterns = [
    path('api/payment/initiate_payment', get_payment, name='get_payment'),
    path('api/payment/get_transaction_info', user_transaction_info, name='get_transaction_info'),
]
```